List in PYTHON

Introduction

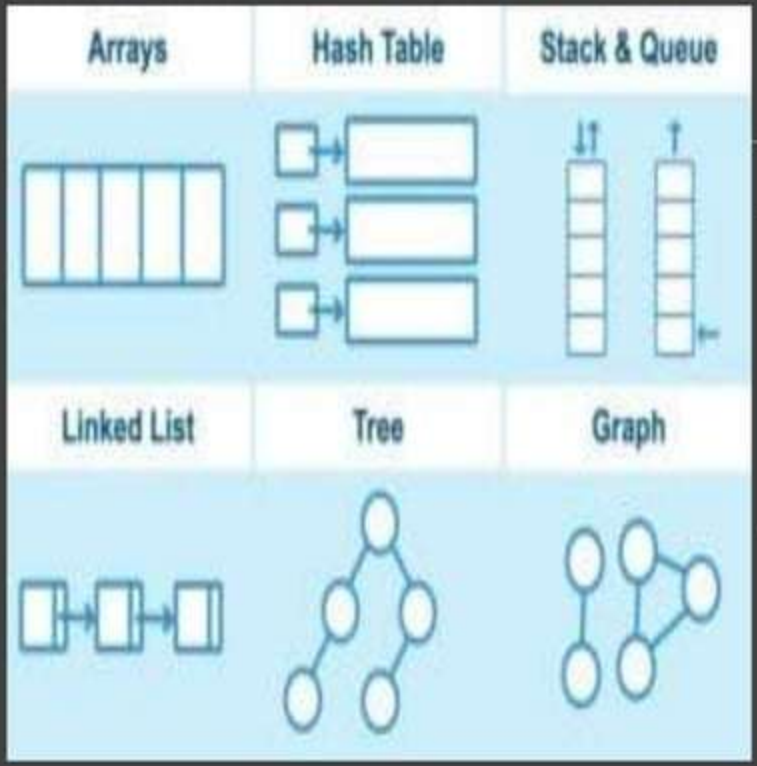
What is Data ?

Data are individual facts, statistics, or items of information, often numeric, that are collected through observation. →Wikipedia

What is Data structure?

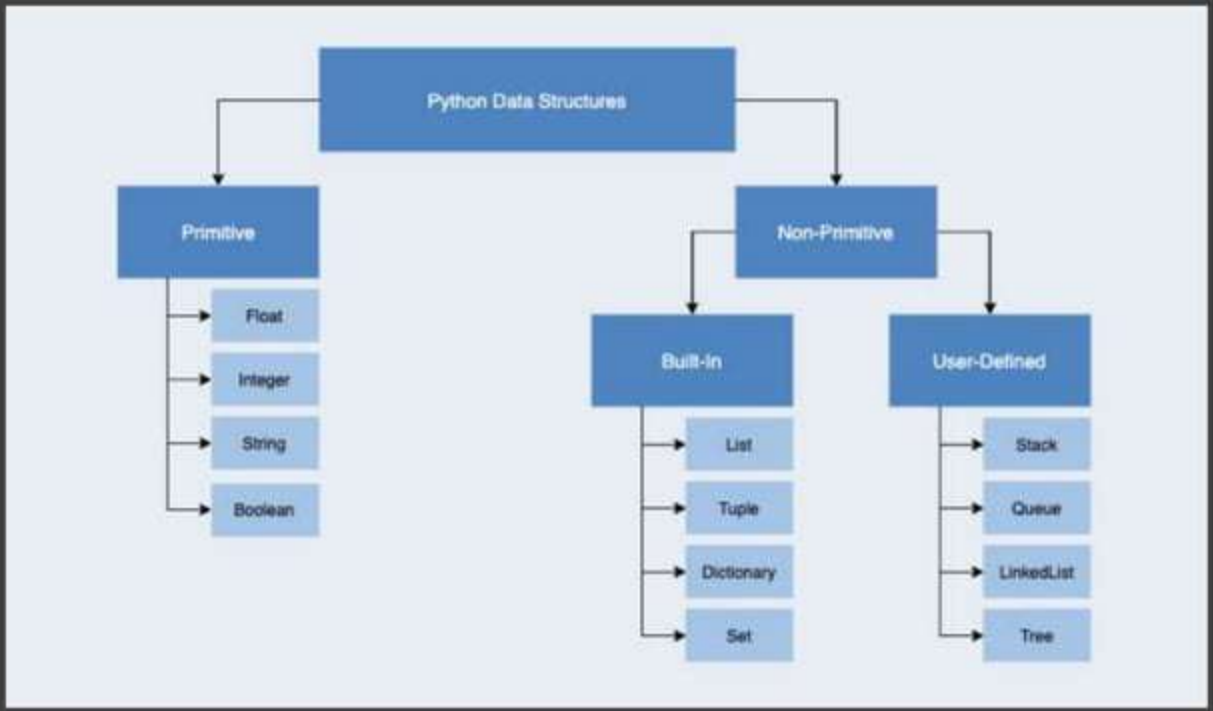
The data structure name indicates itself that organizing the data in memory. There are many ways of organizing the data in the memory

Type	Usage	Examples
int	integer numbers	0 420
double	floating-point numbers	3.14 -200.0
char	characters	'a' '@'
string	sequence of characters	"Hello World!" "Codecademy"
bool	truth values	true false



Files

Python Data Structure



list

Lists are used to store multiple items in a single variable. **Lists in Python** can be created by just placing the sequence of items inside the square brackets[].

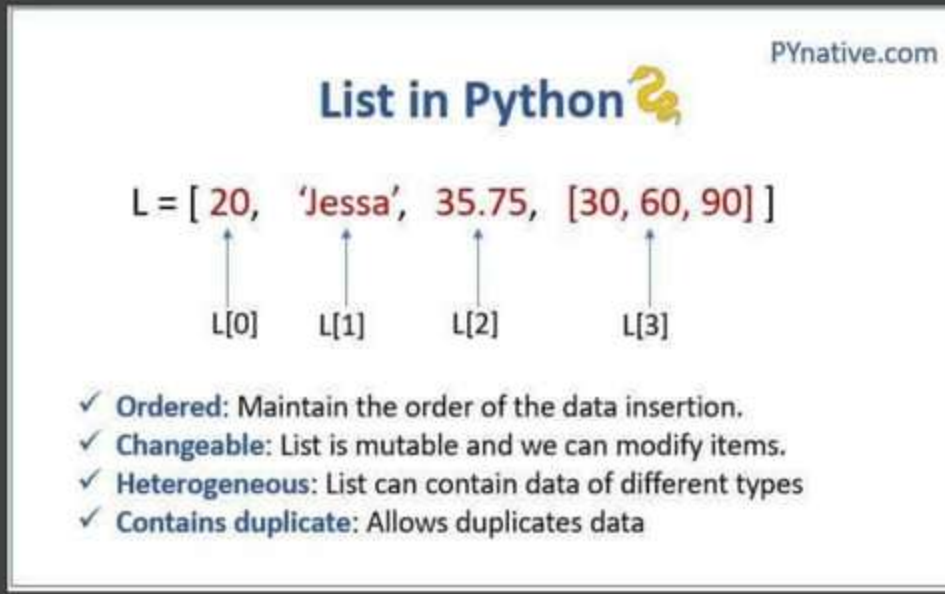
Syntax

Listname = [item1, item2, ……,item n]

Example

mylist = ["apple", "banana", "cherry"]

List in Python



List operation

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

List Items - Data Types

String, int and boolean data types:

Example

list1 = ["apple", "banana", "cherry"]

list2 = [1, 5, 7, 9, 3]

list3 = [True, False, False]

list4 = ["abc", 34, True, 40, "male"]→GUESS???

type()-What is the data type of a list?

From Python's perspective, lists are defined as objects with the data type 'list':

Example

```
mylist = ["apple", "banana", "cherry"]
print(type(mylist))
```

OUTPUT: <class 'list'>

Allow Duplicates- Lists allow duplicate values

Since lists are indexed, lists can have items with the same value:

Example

```
Thislist=["apple", "banana", "cherry", "apple", "cherry"]
print(thislist)
```

OUTPUT: apple, banana, cherry, apple, cherry

List Length-Print the number of items in the list

To determine how many items a list has, use the len() function:

Example

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

OUTPUT: 3

Python List count() Method

The count() method returns the number of elements with the specified value.

Syntax

list.count(*value*)

Example

Return the number of times the value 9 appears int the list:

```
points = [1, 4, 2, 9, 7, 8, 9, 3, 1]
x = points.count(9)
```

OUTPUT: 2

Python List index() Method

The index() method returns the position at the first occurrence of the specified value.

Syntax

list.index(*elmnt*)

Example

What is the position of the value 32:

```
fruits = [4, 55, 64, 32, 16, 32]
x = fruits.index(32)
```

OUTPUT: 3

Access Items-Print the specific item of the list:

List items are indexed and you can access them by referring to the index number:

Example

```
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
```

OUTPUT: banana

Access Items-Negative Indexing

Negative indexing means start from the end

-1 refers to the last item, -2 refers to the second last item etc.

Example

Print the last item of the list:

```
thislist = ["apple", "banana", "cherry"]
print(thislist[-1])
```

OUTPUT: cherry

Access Items-Range of Indexes

You can specify a range of indexes by specifying where to start and where to end the range.

When specifying a range, the return value will be a new list with the specified items.

Example

Return the third, fourth, and fifth item:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:5])
```

OUTPUT:['cherry', 'orange', 'kiwi']

Accessing the list item

List = [0, 1, 2, 3, 4, 5]

0	1	2	3	4	5

List[0] = 0

List[1] = 1

List[2] = 2

List[3] = 3

List[4] = 4

List[5] = 5

List[0:] = [0,1,2,3,4,5]

List[:] = [0,1,2,3,4,5]

List[2:4] = [2, 3]

List[1:3] = [1, 2]

List[:4] = [0, 1, 2, 3]

Access Items-Check if Item Exists

To determine if a specified item is present in a list use the in keyword:

Example

Check if "apple" is present in the list:

```
thislist = ["apple", "banana", "cherry"]
if "apple" in thislist:
print("Yes, 'apple' is in the fruits list")
```

OUTPUT: Yes, 'apple' is in the fruits list

Change List Items

To change the value of a specific item, refer to the index number:

Example

Change the second item:

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"
print(thislist)
```

OUTPUT:[apple, blackcurrant, cherry]

Change List Items- Change a Range of Item Values

To change the value of items within a specific range, define a list with the new values, and refer to the range of index numbers where you want to insert the new values:

Example

Change the values "banana" and "cherry" with the values "blackcurrant" and "watermelon":

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "mango"]
thislist[1:3] = ["blackcurrant", "watermelon"]
print(thislist)
```

OUTPUT:

Guess??

Example

```
thislist = ["apple", "banana", "cherry"]
thislist[1:3] = ["watermelon"]
print(thislist)
```

OUTPUT:

Change List Items- Insert()

To insert a new list item, without replacing any of the existing values, we can use the insert() method.

The insert() method inserts an item at the specified index:

Syntax

```
insert(position, item)
```

Example

```
thislist=["apple", "banana", "cherry"]

thislist.insert(2,"watermelon")
print(thislist)
```

Add List Items- Append Items

To add an item to the end of the list, use the append() method:

Syntax

list.append(*elmnt*)

Example

Using the append() method to append an item:

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

```
OUTPUT:['apple', 'banana', 'cherry', 'orange']
```

Add List Items- Insert Items

To insert a list item at a specified index, use the insert() method.

Example

Insert an item as the second position:

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)
```

OUTPUT:

GUESS??

Add List Items- Extend List

To append elements from *another list* to the current list, use the extend() method.

Syntax

list.extend(*iterable*)

Example

Add the elements of tropical to thislist:

```
thislist = ["apple", "banana", "cherry"]
tropical = ["mango", "pineapple", "papaya"]
thislist.extend(tropical)
print(thislist)
```

OUTPUT:

Remove List Items-Remove Specified Item

The remove() method removes the specified item.

Example

Remove "banana":

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

OUTPUT: [“apple”, “cherry”]

Remove List Items- Remove Specified Index

The pop() method removes the specified index.

Example

Remove the second item:

```
thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)
```

OUTPUT:

Remove List Items- Remove Specified Index(del)

The del keyword also removes the specified index:

Example

Remove the first item:

```
thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
```

OUTPUT:

Guess???

```
thislist = ["apple", "banana", "cherry"]
thislist.pop()
print(thislist)
```

OUTPUT:

```
thislist = ["apple", "banana", "cherry"]
del thislist
```

OUTPUT:

Remove List Items-Clear the List

The clear() method empties the list. The list still remains, but it has no content.

Syntax

list.clear()

Example

Clear the list content:

```
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
```

OUTPUT:[ ]

Loop Through a List

Example

Print all items in the list, one by one:

```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
print(x)
```

For-loop
While-loop
Do-while loop

```
OUTPUT:
apple
banana
cherry
```

Python - Sort Alphanumerically

List objects have a sort() method that will sort the list alphanumerically, ascending, by default:

Example

Sort the list alphabetically:

```
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.sort()
print(thislist)
```

```
OUTPUT:['banana', 'kiwi', 'mango', 'orange', 'pineapple']
```

Python - Sort numerical

Example

Sort the list numerically:

```
thislist = [100, 50, 65, 82, 23]
thislist.sort()
print(thislist)
```

OUTPUT:[23, 50, 65, 82, 100]

```
thislist = [10, "mango", "kiwi",95, "pineapple", "banana",40,67]

thislist.sort()

print(thislist)
```

GUESS

OUTPUT:

Python - Sort Descending

To sort descending, use the keyword argument reverse = True:

Example

Sort the list descending:

```
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.sort(reverse = True)
print(thislist)
```

```
OUTPUT: ['pineapple', 'orange', 'mango', 'kiwi', 'banana']
```

Python - Case Insensitive Sort

By default the sort() method is case sensitive, resulting in all capital letters being sorted before lower case letters:

Example

Case sensitive sorting can give an unexpected result:

```
thislist = ["banana", "Orange", "Kiwi", "cherry"]
thislist.sort()
print(thislist)
```

thislist.sort(key = str.lower)

```
OUTPUT: ['Kiwi', 'Orange', 'banana', 'cherry']
```

Python - Reverse Order

The reverse() method reverses the current sorting order of the elements.

Example

Reverse the order of the list items:

```
thislist = ["banana", "Orange", "Kiwi", "cherry"]
thislist.reverse()
print(thislist)
```

```
OUTPUT: ['cherry', 'Kiwi', 'Orange', 'banana']
```

Python - Copy Lists

You cannot copy a list simply by typing list2 = list1, because: list2 will only be a *reference* to list1, and changes made in list1 will automatically also be made in list2.

There are ways to make a copy, one way is to use the built-in List method copy().

Example

Make a copy of a list with the copy() method:

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist.copy()
print(mylist)
```

OUTPUT: ['apple', 'banana', 'cherry']

Python - Join Lists- Join Two Lists

There are several ways to join, or concatenate, two or more lists in Python.

One of the easiest ways are by using the + operator.

Example

Join two list:

```
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]

list3 = list1 + list2
print(list3)
```

OUTPUT: ['a', 'b', 'c', 1, 2, 3]

Cont...

you can use the extend() method, which purpose is to add elements from one list to another list:

Example

Use the extend() method to add list2 at the end of list1:

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]
list1.extend(list2)
print(list1)
```

OUTPUT: ['a', 'b', 'c', 1, 2, 3]

Cont...

Example

Add a list to a list:

```
a = ["apple", "banana", "cherry"]
b = ["Ford", "BMW", "Volvo"]
a.append(b)
```

OUTPUT: ['apple', 'banana', 'cherry', ["Ford", "BMW", "Volvo"]]

Summary

List is a collection which is ordered and changeable. Allows duplicate members.

Tuple is a collection which is ordered and unchangeable. Allows duplicate members.

Thank you